Stanford Machine Learning — Week 2. § Linear Regression with multiple variables.

- multiple inputs (information) to predict.
- input: $x_1, x_2, \ldots, x_n$. ; output = y.

e.g.

x_1 size	x_2 # of bedrooms	x_3 Number of Flors	x_4 Age of home	price (y)
				} m samples

Notation:

n - number of features.

$x^{(i)}$ - input (features) of i^{th} training sample → $x^{(i)} = \begin{pmatrix} x^{(i)}_1 \\ x^{(i)}_2 \\ \vdots \\ x^{(i)}_n \end{pmatrix}$

(i: index to training set; n-dimensional vector.)

$x^{(i)}_j$ - value of feature j in i^{th} training sample

$\Rightarrow \quad x^{(i)} = [x^{(i)}_j]$.

- Hypothesis

previously $h_\theta(x) = \theta_0 + \theta_1 x$

now → $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$

define: $x_0 = 1$, i.e. $x_0^{(i)} = 1 \quad \forall i$

$$\Rightarrow \quad x = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \; ; \quad \theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}.$$

$$h_\theta(x) = \theta_0 x_0 + \cdots + \theta_n x_n$$

$$= \theta^T \cdot x \quad \text{(expressed as inner product of vectors)}$$

⇒ Multivariate Linear Regression: $h_\theta(x) = \theta^T \cdot x$.

Linear Regression with Multiple Variables - Gradient descent for Multiple Variables

Recall: Hypothesis $h_\theta(x) = \underline{\theta}^T \cdot \underline{x} = \sum \theta_i x_i$

Parameters $\underline{\theta} = \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix} \rightarrow \in \mathbb{R}^{n+1}$.

Cost function. $J(\underline{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} [\, h_\theta(\underline{x}^{(i)}) - y^{(i)} \,]^2$

Gradient descent for multiple variables.

1) repeat {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\underline{\theta})$ [simultaneously update $\forall j = 0 \cdots n$]

}

=> 2) repeat until convergence {

$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} [\, h_\theta(\underline{x}^{(i)}) - y^{(i)} \,] \cdot x_j^{(i)}$

(i denotes sample #; j denotes features)

}

Note: $x_0^{(i)} = 1$. by def'n.

Linear Regression with multiple variables - Gradient descent

Feature Scaling

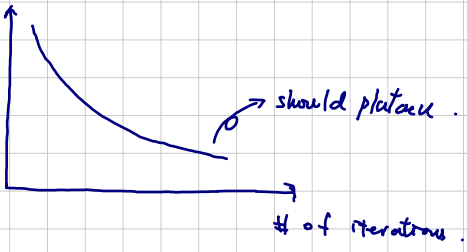
- Idea: make sure features are on similar schedule.
- Get every feature into $-1 \le x_i \le 1$ range. (wonder).

Mean Normalization

- $x_i \rightarrow x_i - \mu_i$ ($E(x_i - \mu_i) = 0$, $x_i - \mu_i$ has zero mean)
- standard normal: $x_i \rightarrow \frac{x_i - \mu_i}{s_i}$
  - ↳ approximate standard deviation by range. (max-min)

Learning Rate

- make sure gradient descent is working → plot. $\min_\theta J_\theta$ over # of iterations.
- for sufficiently small α, J_θ should decrease after every iteration



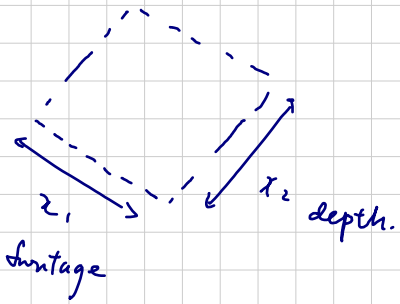
Summary

- α
  - too small: slow convergence
  - too large: may not converge

- to choose α, try 0.001, 0.01, 0.1, 1.

Features and Polynomial Regression

ex. housing price prediction.



$h_\theta(x) = \theta_0 + \theta_1 \times x_1 + \theta_2 \times x_2$; alternatively let x = frontage × depth (area)

let $h_\theta(x) = \theta_0 + \theta_1 \cdot x$

- Polynomial regression: "fitting" into different polynomials. by choice

  - We can approach this using multivariate regression.

    → $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

    $x_1 = x$ $\quad x_2 = x^2$ $\quad x_3 = x^3$

  - note: apply "feature scaling" to each of x_1, x_2, x_3 !

§ Computing Parameters Analytically

Normal Equation

- Gradient descent.
  ↳ iterative approach

- to find min J.
  ↳ set $\frac{\partial}{\partial \theta_j} J(\theta) = 0$; solve for $\theta_0, \theta_1, \ldots \theta_n$.

- Normal method: solving for θ analytically.
- construct matrix. $X = \begin{pmatrix} x_0 & x_1 & \cdots & x_n \end{pmatrix}$ } $n+1$; n features.
  m samples

$$\theta = (X^T X)^{-1} \cdot X^T y.$$

$y = \begin{pmatrix} \vdots \end{pmatrix}$ m dimensional vector

- General case. m examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$; n features

$x^{(i)} = \begin{pmatrix} x_0^{(i)} \\ \vdots \\ x_n^{(i)} \end{pmatrix} \in \mathbb{R}^{n+1}$; Design matrix. $X = \begin{bmatrix} -[x^{(1)}]^T- \\ \vdots \\ -[x^{(m)}]^T- \end{bmatrix} \in \mathbb{R}^{m \times (n+1)}$

e.g. if $x^{(i)} = \begin{pmatrix} 1 \\ x_1^{(i)} \end{pmatrix}$, $X = \begin{pmatrix} 1 & x_1^{(1)} \\ \vdots & \vdots \\ 1 & x_m^{(1)} \end{pmatrix}$; $y = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix}$

$$\theta = (X^T X)^{-1} X^T y$$

$(X^T X)^{-1}$ is inverse of $X^T X$

- **Gradient descent**

- need to chose α
- need many iterations
- work well wl large n. (10^6).

Normal Equation

- need to compute $(X^T X)^{-1}$
  ↓
  slow if n is large. ($O(n^3)$ for inverse computation)

Normal equation and non-invertability

Normal equation: $\theta = (X^T X)^{-1} X^T y$

- What if $X^T X$ is non-invertable.
  - (1) Redudant features (linearly dependent)
  - (2) Too many features ($m \leq n$), more than sets.
    ↳ delete some features, or use regularization. (later)